Cika Electrónica
Nota de Aplicación: CAN-087
Título: **Utilización de displays LCD color con controladores S1D13706 y Rabbit**
Autor: Sergio R. Caprile, Senior Engineer

Revisiones	Fecha	Comentarios
0	04/01/08	

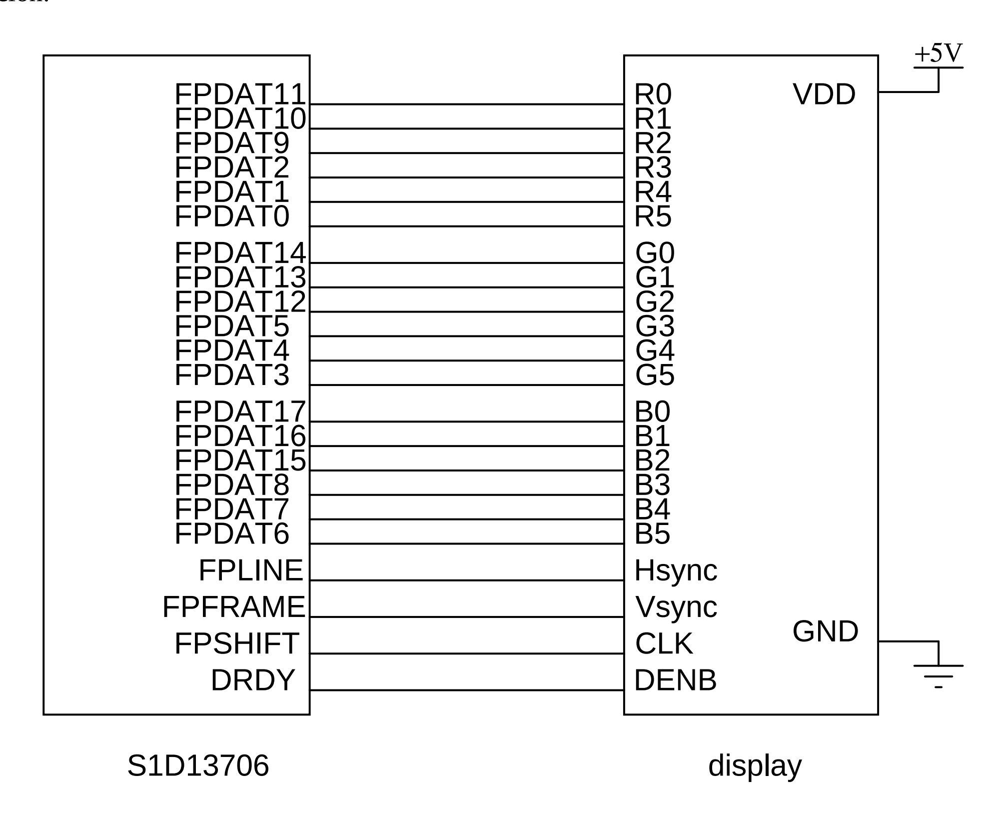
Modificamos levemente el desarrollo de las CAN-035, CAN-036 y CAN-037, para trabajar con displays TFT de 640x480 en formato VGA, como por ejemplo el PD064VT4 presentado en CTC-053.

Limitaciones

Dadas las características de memoria del controlador, no es posible tener más de cuatro colores simultáneamente en pantalla. Sin embargo, cada uno de estos colores se puede obtener de una paleta de 18-bits y ser modificado en tiempo real.

Hardware de display

La conexión al display se realiza mediante las líneas analizadas en CTC-053, como indica el diagrama a continuación:



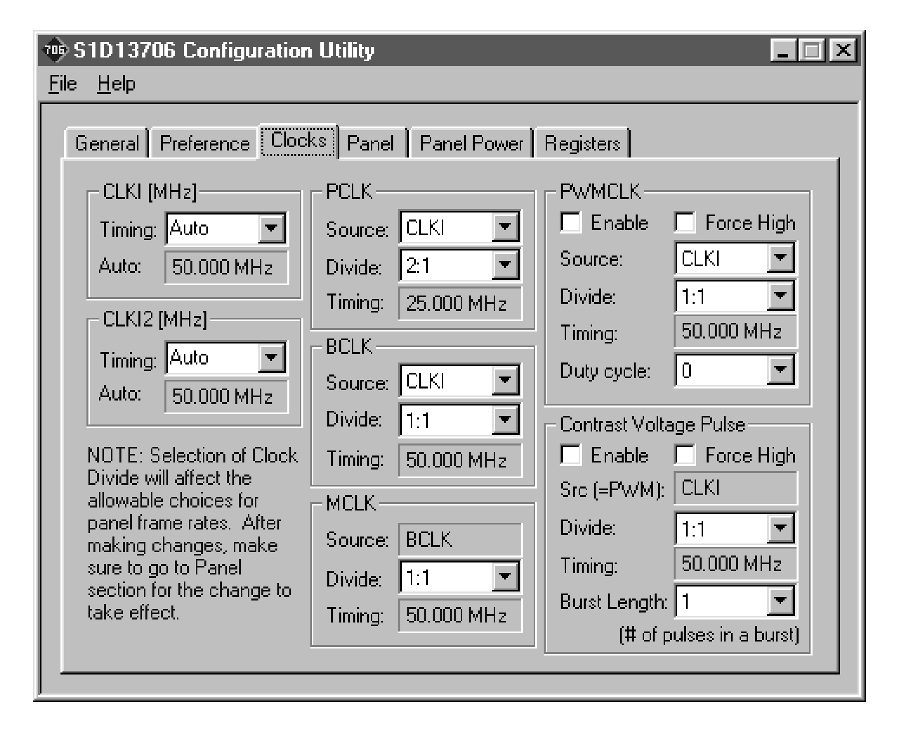
Es fundamental mantener conexiones cortas, no olvidemos que estamos trabajando con un bus de dieciocho señales de 25MHz.

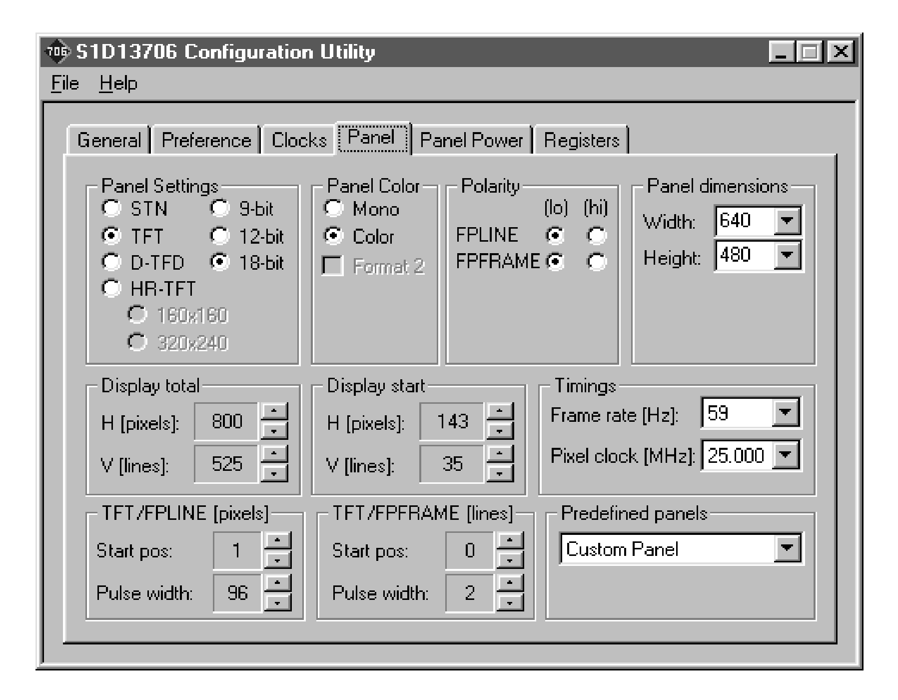
Configuración del S1D13706

Para obtener los valores a setear en cada uno de los registros, utilizamos el software de configuración provisto por el fabricante. Primero deberemos indicar 2bpp en la solapa *Preferences*. A continuación en la solapa *Clocks*, la frecuencia de reloj corresponderá al utilizado, en este caso 50 MHz, el divisor de PCLK debe ser 2:1 para poder lograr el pixel clock de 25MHz. En este desarrollo, utilizamos la entrada CLKI para ingresar el reloj. Luego, en la solapa Panel, definimos uno a uno los parámetros del panel:**TFT de 18-bits, color, 640x480 pixels**. Para generar el timing de VGA, el cual requiere este display, transferimos las especificaciones en microsegundos a unidades en pixel clocks.

El resto de los parámetros podemos dejar los valores por defecto, o leer detenidamente el manual técnico del S1D13706 para saber cómo configurar a nuestro antojo. Exportamos luego los datos en un archivo de tipo *C header file* (s1d13706.h), el cual podremos editar e incluir en el código para Rabbit.

Puede observarse un ejemplo del seteo de estas opciones en las pantallas que figuran a continuación:





Configuración del display

A fin de mantener compatibilidad con lo desarrollado anteriormente, configuramos los dip-switches del PD064VT4 para un shift de 8 pixels y 3 líneas; los switches de barrido los colocaremos 1:off, 2:on dado que no utilizamos los pines correspondientes.

Software de bajo nivel

El hardware y el entorno son muy similares a los de CAN-036, por lo que utilizaremos el mismo set de rutinas. La única diferencia es que tenemos cuatro colores, por lo tanto nuestra paleta tendrá cuatro triplets.

A continuación, la inicialización del chip. Los valores los obtuvimos utilizando el software de configuración provisto por el fabricante, según comentáramos.

```
typedef unsigned short S1D_INDEX;
typedef unsigned char  S1D_VALUE;

typedef struct
{
    S1D_INDEX Index;
    S1D_VALUE Value;
} S1D_REGS;

const static S1D_REGS aS1DRegs[] =
{
    {0x04,0x00},   // BUSCLK MEMCLK Config Register
    {0x05,0x12},   // PCLK Config  Register
    {0x10,0x61},   // PANEL Type Register
    {0x11,0x00},   // MOD Rate Register
    {0x12,0x63},   // Horizontal Total Register
    {0x14,0x4F},   // Horizontal Display Period Register
    {0x16,0x8A},   // Horizontal Display Period Start Pos Register 0
    {0x17,0x00},   // Horizontal Display Period Start Pos Register 1
    {0x18,0x0C},   // Vertical Total Register 0
    {0x19,0x02},   // Vertical Total Register 1
    {0x1C,0xDF},   // Vertical Display Period Register 0
    {0x1D,0x01},   // Vertical Display Period Register 1
    {0x1E,0x23},   // Vertical Display Period Start Pos Register 0
    {0x1F,0x00},   // Vertical Display Period Start Pos Register 1
    {0x20,0x5F},   // Horizontal Sync Pulse Width Register
    {0x22,0x00},   // Horizontal Sync Pulse Start Pos Register 0
```

```
    {0x23,0x00},   // Horizontal Sync Pulse Start Pos Register 1
    {0x24,0x01},   // Vertical Sync Pulse Width Register
    {0x26,0x00},   // Vertical Sync Pulse Start Pos Register 0
    {0x27,0x00},   // Vertical Sync Pulse Start Pos Register 1
    {0x70,0x01},   // Display Mode Register
    {0x71,0x00},   // Special Effects Register
    {0x74,0x00},   // Main Window Display Start Address Register 0
    {0x75,0x00},   // Main Window Display Start Address Register 1
    {0x76,0x00},   // Main Window Display Start Address Register 2
    {0x78,0x28},   // Main Window Address Offset Register 0
    {0x79,0x00},   // Main Window Address Offset Register 1
    {0x7C,0x00},   // Sub Window Display Start Address Register 0
    {0x7D,0x00},   // Sub Window Display Start Address Register 1
    {0x7E,0x00},   // Sub Window Display Start Address Register 2
    {0x80,0x50},   // Sub Window Address Offset Register 0
    {0x81,0x00},   // Sub Window Address Offset Register 1
    {0x84,0x00},   // Sub Window X Start Pos Register 0
    {0x85,0x00},   // Sub Window X Start Pos Register 1
    {0x88,0x00},   // Sub Window Y Start Pos Register 0
    {0x89,0x00},   // Sub Window Y Start Pos Register 1
    {0x8C,0x4F},   // Sub Window X End Pos Register 0
    {0x8D,0x00},   // Sub Window X End Pos Register 1
    {0x90,0xEF},   // Sub Window Y End Pos Register 0
    {0x91,0x00},   // Sub Window Y End Pos Register 1
    {0xA0,0x00},   // Power Save Config Register
    {0xA1,0x00},   // CPU Access Control Register
    {0xA2,0x00},   // Software Reset Register
    {0xA3,0x00},   // BIG Endian Support Register
    {0xA4,0x00},   // Scratch Pad Register 0
    {0xA5,0x00},   // Scratch Pad Register 1
    {0xA8,0x00},   // GPIO Config Register 0
    {0xA9,0x80},   // GPIO Config Register 1
    {0xAC,0x00},   // GPIO Status Control Register 0
    {0xAD,0x00},   // GPIO Status Control Register 1
    {0xB0,0x00},   // PWM CV Clock Control Register
    {0xB1,0x00},   // PWM CV Clock Config Register
    {0xB2,0x00},   // CV Clock Burst Length Register
    {0xB3,0x00},   // PWM Clock Duty Cycle Register
};

#define S1DNUMREGS 54
#define S1DMEMSIZE 76800

void init13706()
{
int i;

       BitWrPortI(PCDR,&PCDRShadow,0,2);                   // M/R=0 => registros
       for(i=1;i<S1DNUMREGS;i++)
       write13706((aS1DRegs[i].Index),aS1DRegs[i].Value);
       writepalette(palette);                              // al regresar, M/R=1 => memoria
}
```

Software

El resto del software lo escribimos mayormente en C, por comodidad y velocidad de desarrollo. Se trata de simples y comunes rutinas que no incluiremos aquí para no extender el texto, pero que el lector puede obtener del archivo adjunto con el software, o consultar en cualquiera de las otras notas de aplicación, dado que son muy similares.

Nota importante

Tanto el fabricante del display como el del controlador recomiendan respetar un ciclo de encendido y apagado para maximizar la vida útil del display. Si el proceso de inicialización del controlador no es lo suficientemente rápido, deberemos desarrollar algún método de control de la alimentación del display